A modification of the PC algorithm yielding order-independent skeletons

Diego Colombo colombo@stat.math.ethz.ch
Marloes H. Maathuis maathuis@stat.math.ethz.ch
Seminar for Statistics,
ETH Zurich,
8092 Zurich, Switzerland

Editor:

Abstract

We consider the PC algorithm for estimating a Markov equivalence class of directed acyclic graphs. This algorithm is known to be order-dependent, in the sense that the output depends on the order in which the variables are given. We show that this order-dependence is not just an aesthetical problem, and can lead to highly variable results in high-dimensional settings. We propose a simple modification, called PC-stable, that yields order-independent adjacencies in its output graph and is consistent in high-dimensional settings under the same conditions as the original PC algorithm. We compare the PC and PC-stable algorithms in simulation studies and on a yeast gene expression data set, and show that PC-stable yields significantly improved performance in high-dimensional settings. The modification used in PC-stable can be easily combined with other adaptations of the PC algorithm, including hybrid algorithms and the FCI algorithm. All software is implemented in the R-package `pcalg`.

Keywords: PC algorithm, order-dependence, skeleton, consistency, high-dimensionality

1. Introduction

We consider the PC algorithm (Spirtes et al., 2000) for learning Markov equivalence classes of directed acyclic graphs (DAGs). The PC algorithm is a so-called constraint-based method, as it relies on conditional independence constraints. Alternative approaches include score-based and Bayesian methods (e.g., Chickering (2002); Cooper and Herskovits (1992); Heckerman et al. (1995); Spiegelhalter et al. (1993)), as well as hybrids of the different approaches (e.g., Dash and Druzdzel (1999); Singh and Valtorta (1993); Spirtes and Meek (1995); Tsamardinos et al. (2006); van Dijk et al. (2003)).

A Markov equivalence class of DAGs can be uniquely described by a so-called CPDAG, which is a graph consisting of directed and/or undirected edges (see Section 2 for a precise definition). The PC algorithm is widely used for learning CPDAGs in high-dimensional settings (e.g., Kalisch et al. (2010); Nagarajan et al. (2010); Stekhoven et al. (2012); Zhang et al. (2011)). An important reason for this is that PC algorithm is computationally feasible for sparse graphs with up to thousands of variables, and open-source software is available (e.g., `pcalg` (Kalisch et al., 2012) and TETRAD IV (Spirtes et al., 2000)). Moreover, the

PC algorithm has been shown to be consistent for high-dimensional sparse graphs under some conditions (Harris and Drton, 2012; Kalisch and Bühlmann, 2007).

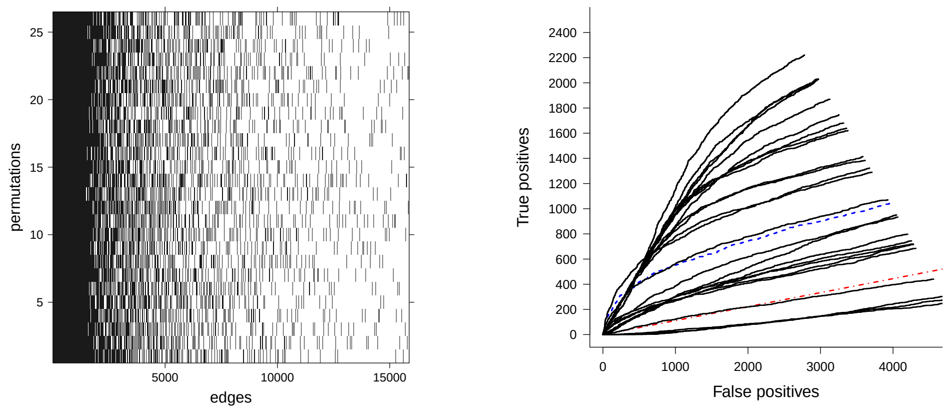
When the PC algorithm is applied to data, it is generally order-dependent, in the sense that its output depends on the order in which the variables are given. Spirtes et al. (2000) (Section 5.4.2.4) exploit this order-dependence to obtain faster algorithms, by removing the "weakest" edges as early as possible. Dash and Druzdzel (1999) exploit the order-dependence for a different purpose, namely to obtain candidate graphs for a score-based approach. Cano et al. (2008) resolve the order-dependence via a rather involved method based on measuring edge strengths. Overall, however, the order-dependence of the PC algorithm has received relatively little attention in the literature, suggesting that it seems to be regarded as a minor issue. We found, however, that the order-dependence can become very problematic for high-dimensional data, leading to highly variable results and conclusions for different orderings on the variables.

In particular, we analyzed a yeast gene expression data set (Hughes et al. (2000); see Section 6 for more detailed information) containing gene expression levels of 5361 genes for 63 wild-type yeast organisms. First, we considered estimating the skeleton of the CPDAG, that is, the undirected graph that results by disregarding all arrowheads in the CPDAG. Figure 1(a) shows the large variability in the estimated skeletons when we use 25 random orderings of the variables. Each estimated skeleton consists of roughly 5000 edges which can be roughly divided into three groups: about 1500 are highly stable and occurred in all orderings, about 1500 are moderately stable and occurred in at least 50% of the orderings, and about 2000 are unstable and occurred in at most 50% of the orderings.

An important motivation for learning DAGs lies in their causal interpretation. We therefore also investigated the effect of different variable orderings on causal inference that is based on the PC algorithm. In particular, we applied the IDA algorithm (Maathuis et al., 2009, 2010) to the yeast gene expression data discussed above. The IDA algorithm conceptually consists of two-steps: first one estimates the Markov equivalence class of DAGs using the PC algorithm, and then one applies Pearl's do-calculus (Pearl, 2000) to each DAG in the Markov equivalence class. (The algorithm uses a fast local implementation that does not require listing all DAGs in the equivalence class.) One can then obtain estimated lower bounds on the sizes of the causal effects between all pairs of genes. For each of the 25 random orderings, we ranked the gene pairs according to these lower bounds, and compared these rankings to a gold standard set of large causal effects computed from gene knock-out data. Figure 1(b) shows the large variability in the resulting receiver operating characteristic (ROC) curves. The ROC curve that was published in Maathuis et al. (2010) was significantly better than random guessing with $p < 0.001$, and is somewhere in the middle. Some of the other curves are much better, while there are also curves that are indistinguishable from random guessing.

The order-dependence in the PC algorithm occurs in two different forms: order-dependence in the skeleton and order-dependence in the edge orientations. Figure 1(a) illustrates that order-dependence in the skeleton can be very large. In this paper, we propose a slight modification in the estimation of the skeleton, which removes this order-dependence.

The remainder of the paper is organized as follows. In Section 2 we discuss some background and terminology. Section 3 explains the original PC algorithm. Section 4 introduces our modification, called PC-stable, yielding order-independent skeletons. PC-stable is iden-



(a) Edges (black) occurring in the estimated skeletons for 25 random variable orderings, as well as for the original ordering (shown as permutation 26). The edges along the x-axis are ordered according to their presence in the output for the 26 permutations, from edges that occurred in all permutations to edges that occurred in only one permutation. Edges that did not occur in any of the permutations were omitted. For technical reasons, only every 10th edge is actually plotted.

(b) ROC curves corresponding to the 25 random orderings of the variables (solid black), where the curves are generated exactly as in Maathuis et al. (2010). The ROC curve for the original ordering of the variables (dashed blue) was published in Maathuis et al. (2010). The dashed-dotted red curve represents random guessing.

Figure 1: Analysis of the yeast gene expression data (Hughes et al., 2000) for 25 random orderings of the variables, using tuning parameter $\alpha = 0.01$. The estimated graphs and resulting causal rankings are highly order-dependent.

tical to the original PC algorithm when perfect conditional independence information is used. When applied to data, PC-stable is consistent in high-dimensional settings under the same conditions as the original PC algorithm. Section 5 compares the PC and PC-stable algorithms in simulations, and Section 6 compares them on the yeast gene expression data discussed above. We show that PC-stable has significantly improved performance for sparse high-dimensional graphs. We close with a discussion in Section 7.

2. Preliminaries

2.1 Graph terminology

A graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ consists of a set of vertices $\mathbf{V} = \{X_1, \ldots, X_p\}$ and a set of edges $\mathbf{E}$. The vertices represent random variables and the edges describe conditional independence and causal relationships.

A graph containing only directed edges ($\rightarrow$) is called *directed*, one containing only undirected edges ($-$) is called *undirected*, and one containing directed and/or undirected edges is called *partially directed*. The *skeleton* of a partially directed graph is the undirected graph that results when all directed edges are replaced by undirected edges.

All graphs we consider are *simple*, meaning that there is at most one edge between any pair of vertices. If an edge is present, the vertices are said to be *adjacent*. If all pairs of vertices in a graph are adjacent, the graph is called *complete*. The *adjacency set* of a vertex X_i in a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, denoted by $\text{adj}(\mathcal{G}, X_i)$, is the set of all vertices in $\mathbf{V}$ that are adjacent to X_i in $\mathcal{G}$. A vertex X_j in $\text{adj}(\mathcal{G}, X_i)$ is called a *parent* of X_i if $X_j \to X_i$. The corresponding set of parents is denoted by $\text{pa}(\mathcal{G}, X_i)$.

A *path* is a sequence of distinct adjacent vertices. A *directed path* is a path along directed edges that follows the direction of the arrowheads. A *directed cycle* is formed by a directed path from X_i to X_j together with the edge $X_j \to X_i$. A (partially) directed graph is called a *(partially) directed acyclic graph ((P)DAG)* if it does not contain directed cycles.

A triple (X_i, X_j, X_k) is called *unshielded* if X_i and X_j as well as X_j and X_k are adjacent, but X_i and X_k are not adjacent. A *v-structure* (X_i, X_j, X_k) is an unshielded triple where the edges as oriented as $X_i \to X_j \leftarrow X_k$.

2.2 Probabilistic and causal interpretation of DAGs

We use the notation $X_i \perp\!\!\!\perp X_j | \mathbf{S}$ to indicate that X_i is independent of X_j given $\mathbf{S}$, where $\mathbf{S}$ is a set of variables not containing X_i and X_j (Dawid, 1980). If $\mathbf{S}$ is the empty set, we simply write $X_i \perp\!\!\!\perp X_j$. If $X_i \perp\!\!\!\perp X_j | \mathbf{S}$, we refer to $\mathbf{S}$ as a *separating set* for (X_i, X_j). A separating set $\mathbf{S}$ for (X_i, X_j) is called *minimal* if there is no proper subset $\mathbf{S}'$ of $\mathbf{S}$ such that $X_i \perp\!\!\!\perp X_j | \mathbf{S}'$.

A distribution Q is said to *factorize* according to a DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ if the the joint density of $\mathbf{V} = (X_1, \dots, X_p)$ can be written as the product of the conditional densities of each variable given its parents in $\mathcal{G}$: $q(X_1, \dots, X_p) = \prod_{i=1}^{p} q(X_i | \text{pa}(\mathcal{G}, X_i))$.

A DAG entails conditional independence relationships via a graphical criterion called *d-separation* (Pearl, 2000). If two vertices X_i and X_j in a DAG $\mathcal{G}$ are d-separated by a subset $\mathbf{S}$ of the remaining vertices, then $X_i \perp\!\!\!\perp X_j | \mathbf{S}$ in any distribution Q that factorizes according to $\mathcal{G}$. A distribution Q is said to be *faithful* to a DAG $\mathcal{G}$ if the reverse implication also holds, that is, if the conditional independence relationships in Q are exactly the same as those that can be inferred from $\mathcal{G}$ using d-separation.

Several DAGs can describe exactly the same conditional independence information. Such DAGs are called Markov equivalent and form a Markov equivalence class. Markov equivalent DAGs have the same skeleton and the same v-structures, and a Markov equivalence class can be described uniquely by a completed partially directed acyclic graph (CPDAG) (Andersson et al., 1997; Chickering, 2002). A CPDAG is a PDAG with the following properties: every directed edge exists in every DAG in the Markov equivalence class, and for every undirected edge $X_i - X_j$ there exists a DAG with $X_i \to X_j$ and a DAG with $X_i \leftarrow X_j$ in the Markov equivalence class. A CPDAG $\mathcal{C}$ is said to *represent* a DAG $\mathcal{G}$ if $\mathcal{G}$ belongs to the Markov equivalence class described by $\mathcal{C}$.

A DAG can be interpreted causally in the following way (Pearl, 2000, 2009; Spirtes et al., 2000): X_1 is a direct cause of X_2 only if $X_1 \to X_2$, and X_1 is a possibly indirect cause of X_2 only if there is a directed path from X_1 to X_2.

Under the assumptions of faithfulness and causal sufficiency (no hidden confounders and no hidden selection variables), the PC algorithm can be used to learn causal CPDAGs from conditional independence information. The FCI algorithm Spirtes et al. (2000) is designed for the analogous problem without causal sufficiency. Several adaptations of the FCI algorithm, including the fast RFCI algorithm, can be found in Colombo et al. (2012).

3. The PC algorithm

We now describe the PC algorithm in detail. In Section 3.1, we first discuss the algorithm under the assumption that we have perfect conditional independence information between all variables in $\mathbf{V}$. We refer to this as the *oracle version*. In Section 3.2 we discuss the more realistic situation where conditional independence relationships have to be estimated from data. We refer to this as the *sample version*.

3.1 Oracle version

A sketch of the PC algorithm is given in Algorithm 3.1. We see that the algorithm consists of three steps. Step 1 finds the skeleton and separation sets, while Steps 2 and 3 determine the orientations of the edges.

Algorithm 3.1 The PC algorithm (oracle version)

```
Require: Conditional independence information among all variables in $\mathbf{V}$, and an ordering
    order($\mathbf{V}$) on the variables
1: Find the skeleton and separation sets using Algorithm 3.2;
2: Orient unshielded triples in the skeleton based on the separation sets;
3: Orient as many of the remaining undirected edges as possible by repeated application
   of rules R1-R3 (see text);
4: return Output graph ($\mathcal{C}$) and separation sets (sepset).
```

Step 1 is given in pseudocode in Algorithm 3.2. We start with a complete undirected graph $\mathcal{C}$. This graph is subsequently thinned out in the loop on lines 3-15 in Algorithm 3.2, where an edge $X_i - X_j$ is deleted if $X_i \perp\!\!\!\perp X_j|\mathbf{S}$ for some subset $\mathbf{S}$ of the remaining variables. These conditional independence queries are organized in a way that makes the algorithm computationally efficient for high-dimensional sparse graphs.

First, when $\ell = 0$, all pairs of vertices are tested for marginal independence. If $X_i \perp\!\!\!\perp X_j$, then the edge $X_i - X_j$ is deleted and the empty set is saved as separation set in sepset(X_i, X_j) and sepset(X_j, X_i). After all pairs of vertices have been considered (and many edges might have been deleted), the algorithm proceeds to the next step with $\ell = 1$.

When $\ell = 1$, the algorithm chooses an ordered pair of vertices (X_i, X_j) still adjacent in $\mathcal{C}$, and checks $X_i \perp\!\!\!\perp X_j|\mathbf{S}$ for subsets $\mathbf{S}$ of size $\ell = 1$ of adj$(\mathcal{C}, X_i) \setminus \{X_j\}$ and of adj$(\mathcal{C}, X_j) \setminus \{X_i\}$. If such a conditional independence is found, the edge $X_i - X_j$ is removed, and the corresponding conditioning set $\mathbf{S}$ is saved in sepset(X_i, X_j) and sepset(X_j, X_i). If all pairs of adjacent vertices have been considered for conditional independence given all subsets of size ℓ of their adjacency sets, the algorithm again increases ℓ by one. This process continues until all adjacency sets in the current graph are smaller than ℓ. At this point the skeleton and the separation sets have been determined.

Algorithm 3.2 Step 1 of the PC algorithm (oracle version)

```
Require: Conditional independence information among all variables in $\mathbf{V}$, and an ordering
    order($\mathbf{V}$) on the variables
1: Form the complete undirected graph $\mathcal{C}$ on the vertex set $\mathbf{V}$
2: Let $\ell = -1$;
3: repeat
4:   Let $\ell = \ell + 1$;
5:   repeat
6:     Select a (new) ordered pair of vertices $(X_i, X_j)$ that are adjacent in $\mathcal{C}$ and satisfy
       $|\text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}| \geq \ell$, using order($\mathbf{V}$);
7:     repeat
8:       Choose a (new) set $\mathbf{S} \subseteq \text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$, using order($\mathbf{V}$);
9:       if $X_i$ and $X_j$ are conditionally independent given $\mathbf{S}$ then
10:        Delete edge $X_i - X_j$ from $\mathcal{C}$;
11:        Let sepset$(X_i, X_j)$ = sepset$(X_j, X_i) = \mathbf{S}$;
12:      end if
13:    until $X_i$ and $X_j$ are no longer adjacent in $\mathcal{C}$ or all $\mathbf{S} \subseteq \text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}$ with
       $|\mathbf{S}| = \ell$ have been considered
14:  until all ordered pairs of adjacent vertices $(X_i, X_j)$ in $\mathcal{C}$ with $|\text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}| \geq \ell$
     have been considered
15: until all pairs of adjacent vertices $(X_i, X_j)$ in $\mathcal{C}$ satisfy $|\text{adj}(\mathcal{C}, X_i) \setminus \{X_j\}| \leq \ell$
16: return $\mathcal{C}$, sepset.
```

Step 2 determines the v-structures. In particular, it considers all unshielded triples in $\mathcal{C}$, and orients an unshielded triple (X_i, X_j, X_k) as a v-structure if and only if $X_j \notin$ sepset(X_i, X_k).

Finally, Step 3 orients as many of the remaining undirected edges as possible by repeated application of the following three rules:

R1: orient $X_j - X_k$ into $X_j \rightarrow X_k$ whenever there is a directed edge $X_i \rightarrow X_j$ such that X_i and X_k are not adjacent (otherwise a new v-structure is created);

R2: orient $X_i - X_j$ into $X_i \rightarrow X_j$ whenever there is a chain $X_i \rightarrow X_k \rightarrow X_j$ (otherwise a directed cycle is created);

R3: orient $X_i - X_j$ into $X_i \rightarrow X_j$ whenever there are two chains $X_i - X_k \rightarrow X_j$ and $X_i - X_l \rightarrow X_j$ such that X_k and X_l are not adjacent (otherwise a new v-structure or a directed cycle is created).

The PC algorithm was shown to be sound and complete.

Theorem 1 *(Theorem 5.1 on p.410 of Spirtes et al. (2000)) Let the distribution of* $\mathbf{V}$ *be faithful to a DAG* $\mathcal{G} = (V, E)$*, and assume that we are given perfect conditional independence information about all pairs of variables* (X_i, X_j) *in* $\mathbf{V}$ *given subsets* $\mathbf{S} \subseteq \mathbf{V} \setminus \{X_i, X_j\}$*. Then the output of the PC algorithm is the CPDAG that represents* $\mathcal{G}$*.*

We briefly discuss the main ingredients of the proof, as these will be useful for understanding our modification in Section 4. The faithfulness assumption implies that conditional independence in the distribution of $\mathbf{V}$ is equivalent to d-separation in the graph $\mathcal{G}$. The skeleton of $\mathcal{G}$ can then be determined as follows: X_i and X_j are adjacent in $\mathcal{G}$ if and only if X_i and X_j are conditionally dependent given any subset $\mathbf{S}$ of the remaining nodes. Naively, one could therefore check all these conditional dependencies, which is known as the SGS algorithm (Spirtes et al., 2000). The PC algorithm obtains the same result with fewer tests, by using the following fact about DAGs: two variables X_i and X_j in a DAG $\mathcal{G}$ are d-separated by some subset $\mathbf{S}$ of the remaining variables if and only if they are d-separated by $\mathrm{pa}(\mathcal{G}, X_i)$ or $\mathrm{pa}(\mathcal{G}, X_j)$. The PC algorithm is guaranteed to check these conditional independencies: at any point in the algorithm the graph $\mathcal{C}$ is a supergraph of the true CPDAG, and the algorithm checks conditional dependencies given all subsets of the adjacency sets, which obviously include the parent sets.

The v-structures are determined based on Lemmas 5.1.2 and 5.1.3 of Spirtes et al. (2000). The soundness and completeness of the orientation rules in Step 3 was shown in Meek (1995).

3.2 Sample version

In applications, we of course do not have perfect conditional independence information. Instead, we assume that we have an i.i.d. sample of size n of $\mathbf{V} = (X_1, \ldots, X_p)$. The conditional independence relationships have to be estimated from these data.

For example, if the distribution of $\mathbf{V}$ is multivariate Gaussian, one can test conditional independence by testing for zero partial correlation. Let $\rho_{n;i,j|\mathbf{S}}$ be the partial correlation between X_i and X_j in $\mathbf{V}$ given a set $\mathbf{S} \subseteq \mathbf{V} \setminus \{X_i, X_j\}$, let $\hat{\rho}_{n;i,j|\mathbf{S}}$ be the corresponding sample correlation, and let $g(x) = \frac{1}{2}\log\left(\frac{1+x}{1-x}\right)$ denote Fisher's z-transform. One can then consider

$$\hat{z}_{n;i,j|\mathbf{S}} = g(\hat{\rho}_{n;i,j|\mathbf{S}}) \quad \text{and} \quad z_{n;i,j|\mathbf{S}} = g(\rho_{n;i,j|\mathbf{S}})$$

and reject the null-hypothesis $H_0(i,j|\mathbf{S}) : \rho_{i,j|\mathbf{S}} = 0$ against the two-sided alternative $H_A(i,j|\mathbf{S}) : \rho_{i,j|\mathbf{S}} \neq 0$ at significance level α if

$$|\hat{z}_{n;i,j|\mathbf{S}}| > (n - |\mathbf{S}| - 3)^{-1/2}\Phi^{-1}(1 - \alpha/2),$$

where $\Phi(\cdot)$ denotes the cumulative distribution function of a standard normal distribution, and we assume $n - |\mathbf{S}| - 3 > 0$.

A sample version of the PC algorithm can be obtained by replacing all steps where conditional independence decisions were taken by such statistical tests. We note that the parameter α is used for many tests, and plays the role of a tuning parameter, where smaller values of α tend to lead to sparser graphs.

3.3 Order-dependence in the sample version

We first consider the role of order($\mathbf{V}$) in Step 1. At each level of ℓ, order($\mathbf{V}$) determines the order in which pairs of adjacent vertices (line 6 in Algorithm 3.2) and subsets $\mathbf{S}$ of their adjacency sets (line 8 in Algorithm 3.2) are considered. We assume that both of these tasks are performed according to the lexicographical ordering of order($\mathbf{V}$), which is the standard implementation in `pcalg` (Kalisch et al., 2012) and TETRAD IV (Spirtes et al., 2000), and is called "PC-1" in Spirtes et al. (2000) (Section 5.4.2.4).

The skeleton $\mathcal{C}$ is updated after each edge removal. Hence, the adjacency sets typically change within one level of ℓ, and this affects which conditional independencies are checked, as the algorithm only conditions on subsets of the adjacency sets.

In the oracle version, we have perfect conditional independence information, and all orderings on the variables lead to the same output. In the sample version, however, we typically make mistakes in keeping or removing edges. In such cases, the resulting changes in the adjacency sets can lead to different outputs. This is illustrated in Example 1.

Example 1 *Suppose that the distribution of $\mathbf{V} = \{X_1, \dots, X_5\}$ is faithful to the DAG in Figure 2(a). This DAG encodes the following conditional independencies with minimal separating sets: $X_1 \perp\!\!\!\perp X_2$ and $X_2 \perp\!\!\!\perp X_4 | \{X_1, X_3\}$.*

Suppose that we have an i.i.d. sample of $\{X_1, \dots, X_5\}$, and that the following conditional independencies with minimal separating sets are judged to hold at some level α: $X_1 \perp\!\!\!\perp X_2$, $X_2 \perp\!\!\!\perp X_4 | \{X_1, X_3\}$, and $X_3 \perp\!\!\!\perp X_4 | \{X_1, X_5\}$. Thus, the first two are correct, while the third is false.

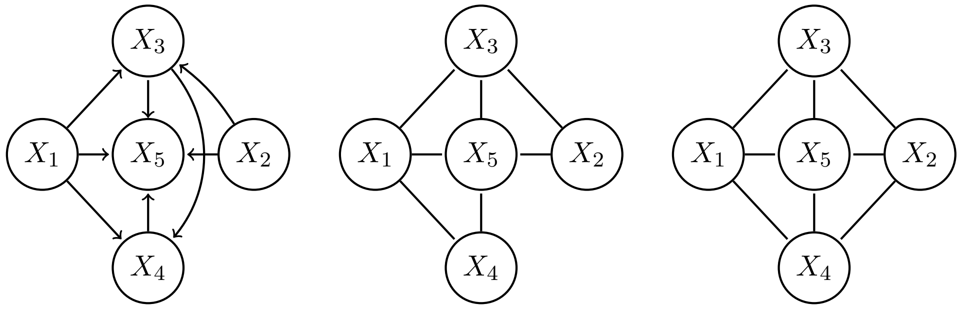
We now apply the PC algorithm with two different orderings: order$_1(\mathbf{V}) = (X_1, X_2, X_3, X_4, X_5)$ and order$_2(\mathbf{V}) = (X_1, X_3, X_2, X_4, X_5)$. The resulting skeletons are shown in Figures 2(b) and 2(c), respectively. We see that the skeletons are different, and that both are incorrect as the edge $X_3 - X_4$ is missing. The skeleton for order$_2(\mathbf{V})$ contains an additional error, as there is an additional edge $X_2 - X_4$.

We now go through Algorithm 3.2 to see what happened. We start with a complete undirected graph on $\mathbf{V}$. When $\ell = 0$, variables are tested for marginal independence, and the algorithm correctly removes the edge between X_1 and X_2. No other conditional independencies are found when $\ell = 0$ or $\ell = 1$. When $\ell = 2$, there are two pairs of vertices that are thought to be conditionally independent given a subset of size 2, namely the pairs (X_2, X_4) and (X_3, X_4).

In order$_1(\mathbf{V})$, the pair (X_2, X_4) is considered first. The corresponding edge is removed, as $X_2 \perp\!\!\!\perp X_4 | \{X_1, X_3\}$ and $\{X_1, X_3\}$ is a subset of adj$(\mathcal{C}, X_4) = \{X_1, X_2, X_3, X_5\}$. Next, the edge between (X_3, X_4) is erroneously removed, because of the wrong decision that $X_3 \perp\!\!\!\perp X_4 | \{X_1, X_5\}$ and the fact that $\{X_1, X_5\}$ is a subset of adj$(\mathcal{C}, X_3) = \{X_1, X_2, X_4, X_5\}$ and adj$(\mathcal{C}, X_4) = \{X_1, X_3, X_5\}$.

In order$_2(\mathbf{V})$, the pair (X_3, X_4) is considered first, and the edge between X_3 and X_4 is erroneously removed. Next, the algorithm considers the pair (X_2, X_4). Since adj$(\mathcal{C}, X_2) = \{X_3, X_4, X_5\}$ and adj$(\mathcal{C}, X_4) = \{X_1, X_2, X_5\}$, the separating set $\{X_1, X_3\}$ of (X_2, X_4) is not a subset of one of these adjacency sets. Therefore, the edge $X_2 - X_4$ remains. In other words, since (X_3, X_4) was considered first in order$_2(\mathbf{V})$, the adjacency set of X_4 was affected and no longer contained X_3, so that the algorithm "forgot" to check the conditional independence $X_1 \perp\!\!\!\perp X_4 | \{X_1, X_3\}$.

The sample version of the PC algorithm is also order-dependent in Steps 2 and 3, where different orderings can lead to different (and sometimes incorrect) separating sets and hence to different v-structures and edge orientations. This is illustrated in Example 3 in the next Section.



(a) True DAG.

(b) Skeleton returned by the sample version of Algorithm 3.2 with order$_1(\mathbf{V})$ and Algorithm 4.2 with any ordering.

(c) Skeleton returned by the sample version of Algorithm 3.2 with order$_2(\mathbf{V})$.

Figure 2: Graphs corresponding to Examples 1 and 2.

4. The PC-stable algorithm

4.1 Oracle version

We now propose a modification of the PC algorithm, called PC-stable. The pseudocode is given in Algorithm 4.1, where the only change is in Step 1 of the algorithm. The pseudocode of the modified Step 1 is given in Algorithm 4.2.

The main difference between Algorithms 3.2 and 4.2 is given by the for-loop on lines 6-8 in the latter one, which computes and stores the adjacency sets $\mathrm{a}(X_i)$ of all variables after each new size ℓ of the conditioning sets. These stored adjacency sets $\mathrm{a}(X_i)$ are used whenever we search for conditioning sets of this given size ℓ. Consequently, an edge deletion on line 13 does no longer affect which conditional independencies are checked for other pairs of variables at this level of ℓ, making the algorithm order-independent in the sample version (see Theorem 3), while remaining sound and complete in the oracle version (see Theorem 2).

In other words, at each level of ℓ, Algorithm 4.2 records which edges should be removed, but for the purpose of the adjacency sets it removes these edges only when it goes to the next value of ℓ. In this sense our approach is opposite to the one of Spirtes et al. (2000) (Section 5.4.2.4), where edges are removed as early as possible.

Theorem 2 *Let the distribution of* $\mathbf{V}$ *be faithful to a DAG* $\mathcal{G} = (V, E)$*, and assume that we are given perfect conditional independence information about all pairs of variables* (X_i, X_j) *in* $\mathbf{V}$ *given subsets* $\mathbf{S} \subseteq \mathbf{V} \setminus \{X_i, X_j\}$*. Then the output of the PC-stable algorithm is the CPDAG that represents* $\mathcal{G}$*.*

The proof of Theorem 2 is analogous to the proof of Theorem 1 for the original PC algorithm, as discussed in Section 3.1.

Algorithm 4.1 The PC-stable algorithm (oracle version)

```
Require: Conditional independence information among all variables in $\mathbf{V}$, and an ordering
    order($\mathbf{V}$) on the variables
 1: Find the skeleton and separation sets using Algorithm 4.2;
 2: Orient unshielded triples in the skeleton based on the separation sets;
 3: Orient as many of the remaining undirected edges as possible by repeated application
    of rules R1-R3 (see text);
 4: return  Output graph ($\mathcal{C}$) and separation sets (sepset).
```

Algorithm 4.2 Step 1 of the PC-stable algorithm (oracle version)

```
Require: Conditional independence information among all variables in $\mathbf{V}$, and an ordering
    order($\mathbf{V}$) on the variables
 1: Form the complete undirected graph $\mathcal{C}$ on the vertex set $\mathbf{V}$
 2: Let $\ell = -1$;
 3: repeat
 4:   Let $\ell = \ell + 1$;
 5:   repeat
 6:     for all vertices $X_i$ in $\mathcal{C}$ do
 7:       Let $\mathrm{a}(X_i) = \mathrm{adj}(\mathcal{C}, X_i)$
 8:     end for
 9:     Select a (new) ordered pair of vertices $(X_i, X_j)$ that are adjacent in $\mathcal{C}$ and satisfy
        $|\mathrm{a}(X_i) \setminus \{X_j\}| \geq \ell$, using order($\mathbf{V}$);
10:     repeat
11:       Choose a (new) set $\mathbf{S} \subseteq \mathrm{a}(X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$, using order($\mathbf{V}$);
12:       if $X_i$ and $X_j$ are conditionally independent given $\mathbf{S}$ then
13:         Delete edge $X_i - X_j$ from $\mathcal{C}$;
14:         Let $\mathrm{sepset}(X_i, X_j) = \mathrm{sepset}(X_j, X_i) = \mathbf{S}$;
15:       end if
16:     until $X_i$ and $X_j$ are no longer adjacent in $\mathcal{C}$ or all $\mathbf{S} \subseteq \mathrm{a}(X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$
        have been considered
17:   until all ordered pairs of adjacent vertices $(X_i, X_j)$ in $\mathcal{C}$ with $|\mathrm{a}(X_i) \setminus \{X_j\}| \geq \ell$ have
      been considered
18: until all pairs of adjacent vertices $(X_i, X_j)$ in $\mathcal{C}$ satisfy $|\mathrm{a}(X_i) \setminus \{X_j\}| \leq \ell$
19: return  $\mathcal{C}$, sepset.
```

4.2 Order-independent skeletons in the sample version

The sample version of the PC-stable algorithm can be obtained as before, by replacing conditional independence queries by conditional independence tests (see Section 3.2). Theorem 3 shows that the sample version of PC-stable yields order-independent skeletons.

Theorem 3 *The skeleton resulting from the sample version of the PC-stable algorithm is order-independent.*

Proof We consider the removal or retention of an arbitrary edge $X_i - X_j$ at some level ℓ.

The ordering of the variables determines the order in which the edges (line 9 of Algorithm 4.2) and the subsets $\mathbf{S}$ of $a(X_i)$ and $a(X_j)$ (line 11 of Algorithm 4.2) are considered. By construction, however, the order in which edges are considered does not affect the sets $a(X_i)$ and $a(X_j)$.

If there is at least one subset $\mathbf{S}$ of $a(X_i)$ or $a(X_j)$ such that $X_i \perp\!\!\!\perp X_j|\mathbf{S}$, then any ordering of the variables will find a separating set for X_i and X_j (but different orderings may lead to different separating sets). Conversely, if there is no subset $\mathbf{S}'$ of $a(X_i)$ or $a(X_j)$ such that $X_i \perp\!\!\!\perp X_j|\mathbf{S}'$, then no ordering will find a separating set.

Hence, any ordering of the variables leads to the same edge deletions, and therefore to the same skeleton. ■

We now illustrate the PC-stable algorithm on the setting of Example 1.

Example 2 *We consider the sample version of the PC-stable algorithm, using as input the same sample conditional independencies and variable orderings as in Example 1. The algorithm now outputs the skeleton shown in Figure 2(b) for both orderings.*

We again go through the algorithm to see what happens. We start with a complete undirected graph on $\mathbf{V}$*. The only conditional independence found when* $\ell = 0$ *or* $\ell = 1$ *is* $X_1 \perp\!\!\!\perp X_2$*, and the corresponding edge is removed. When* $\ell = 2$*, the algorithm first computes the new adjacency sets:* $a(X_1) = a(X_2) = \{X_3, X_4, X_5\}$ *and* $a(X_i) = \mathbf{V} \setminus \{X_i\}$ *for* $i = 3, 4, 5$*. There are two pairs of variables that are thought to be conditionally independent given a subset of size 2, namely* (X_2, X_4) *and* (X_3, X_4)*. Since the sets* $a(X_i)$ *are not updated after edge removals, it does not matter in which order we consider these pairs. Both orderings lead to the removal of both edges, as the separating set* $\{X_1, X_3\}$ *for* (X_2, X_4) *is contained in* $a(X_4)$*, and the separating set* $\{X_1, X_5\}$ *for* (X_3, X_4) *is contained in both* $a(X_3)$ *and* $a(X_4)$*.*

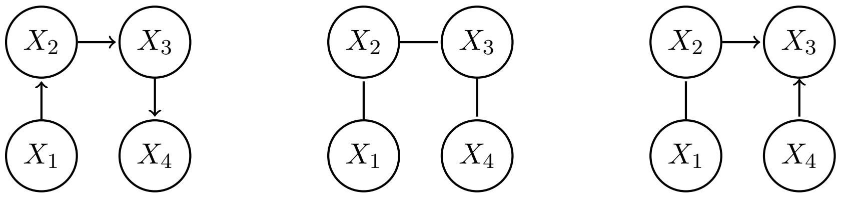
The proof of Theorem 3 also shows that the separating sets resulting from the oracle and sample versions of the PC-stable algorithm are generally still order-dependent. In the sample version this can lead to order-dependence edge-orientations. We illustrate this in Example 3.

Example 3 *Suppose that the distribution of* $\mathbf{V} = \{X_1, X_2, X_3, X_4\}$ *is faithful to the DAG in Figure 3(a). This DAG encodes the following conditional independencies with minimal separating sets:* $X_1 \perp\!\!\!\perp X_3|X_2$*,* $X_2 \perp\!\!\!\perp X_4|X_3$*,* $X_1 \perp\!\!\!\perp X_4|X_2$*, and* $X_1 \perp\!\!\!\perp X_4|X_3$*.*

We first consider the oracle PC and PC-stable algorithms (both give identical results) with two different orderings on the variables: $order_3(\mathbf{V}) = (X_1, X_2, X_3, X_4)$ *and* $order_4(\mathbf{V}) = (X_3, X_2, X_1, X_4)$*. For* $order_3(\mathbf{V})$*, we obtain* $sepset(X_1, X_4) = \{X_2\}$*, while for* $order_4(\mathbf{V})$ *we get* $sepset(X_1, X_4) = \{X_3\}$*. Thus, the information in the separating sets is order-dependent. However, since both versions of the separating sets are correct, they both lead to the same edge orientations: neither one of the unshielded triples* (X_1, X_2, X_3) *and* (X_2, X_3, X_4) *is oriented as a v-structure, so that none of the edges can be oriented.*

Now suppose that we have an i.i.d. sample of (X_1, X_2, X_3, X_4)*. Suppose that at some level* α*, all true conditional independencies are judged to hold, and* $X_2 \perp\!\!\!\perp X_4|X_1$ *is thought to hold by mistake. Now with* $order_3(\mathbf{V})$*, we obtain the incorrect separating set* $sepset(X_2, X_4) =$

$\{X_1\}$, while with order$_4(\mathbf{V})$ we obtain the correct separating set sepset$(X_2, X_4) = \{X_3\}$. So with order$_3(\mathbf{V})$, we incorrectly orient the unshielded triple (X_2, X_3, X_4) as a v-structure (see Figure 3(c)), while order$_4(\mathbf{V})$ correctly leaves the edges undirected (see Figure 3(b)). This illustrates that order-dependent separating sets may lead to order-dependent edge orientations in the sample versions of the PC and PC-stable algorithms.



(a) True DAG.

(b) Output of the oracle PC(-stable) algorithm with any ordering, and of the sample PC(-stable) algorithm with order$_4(\mathbf{V})$.

(c) Output of the sample PC(-stable) algorithm with order$_3(\mathbf{V})$.

Figure 3: Graphs corresponding to Example 3.

4.3 High-dimensional consistency

The original PC algorithm has been shown to be consistent for certain sparse high-dimensional graphs. In particular, Kalisch and Bühlmann (2007) proved consistency for multivariate Gaussian distributions, when using the partial correlation tests discussed in Section 3.2. More recently, Harris and Drton (2012) showed consistency for the broader class of Gaussian copulas when using rank correlations, under slightly different conditions.

These high-dimensional consistency results allow the DAG $\mathcal{G}$ and the number of observed variables p in $\mathbf{V}$ grow as a function of the sample size, so that $p = p_n$, $\mathbf{V} = \mathbf{V}_n = (X_{n,1}, \dots, X_{n,p_n})$ and $\mathcal{G} = \mathcal{G}_n$. The corresponding CPDAGs that represent $\mathcal{G}_n$ are denoted by $\mathcal{C}_n$, and the estimated CPDAGs using tuning parameter α_n are denoted by $\hat{\mathcal{C}}_n(\alpha_n)$. Then the consistency results say that, under some conditions, there exists a sequence α_n such that $P(\hat{\mathcal{C}}_n(\alpha_n) = \mathcal{C}_n) \to 1$ as $n \to \infty$.

These consistency results rely on the fact that the PC algorithm only performs conditional independence tests between pairs of variables given subsets $\mathbf{S}$ of size less than or equal to the degree of the graph (when no errors are made). Our modification still obeys this property, and therefore the consistency results of Kalisch and Bühlmann (2007) and Harris and Drton (2012) remain valid for the PC-stable algorithm, under exactly the same conditions as for the original PC algorithm.

5. Simulations

We compared the estimation performance and computing time of the PC and PC-stable algorithms using simulated data.

5.1 Simulation set-up

We used the following procedure to generate a random DAG with a given number of vertices p and expected neighborhood size $E(N)$. First, we generated a random adjacency matrix A with independent realizations of Bernoulli($E(N)/(p-1)$) random variables in the lower triangle of the matrix and zeroes in the remaining entries. Next, we replaced the ones in A by independent realizations of a Uniform($[0.1, 1]$) random variable. A nonzero entry A_{ij} can be interpreted as an edge from X_j to X_i with "strength" A_{ij}, in the sense that $X_1, \dots, X_p$ can be generated as follows: $X_1 = \epsilon_1$ and $X_i = \sum_{r=1}^{i-1} A_{ir}X_r + \epsilon_i$ for $i = 2, \dots, p$, where $\epsilon_1, \dots, \epsilon_p$ are mutually independent $\mathcal{N}(0, 1)$ random variables. The variables $X_1, \dots, X_p$ then have a multivariate Gaussian distribution with mean zero and covariance matrix $\Sigma = (\mathbf{1} - A)^{-1}(\mathbf{1} - A)^{-T}$, where $\mathbf{1}$ is the $p \times p$ identity matrix.

We used a low-dimensional and a high-dimensional setting. For the low-dimensional setting we generated 250 random DAGs with $p = 50$ and $E(N) = 2$, and for each DAG we generated an i.i.d. sample of size $n = 1000$. For the high-dimensional setting we generated 250 random DAGs with $p = 1000$ and $E(N) = 2$, and for each DAG we generated an i.i.d. sample of size $n = 50$.

We estimated each graph for 50 random orderings of the variables, using the sample versions of the PC and PC-stable algorithms. We used the partial correlation tests discussed in Section 3.2 at level α, where we took $\alpha \in \{0.000625, 0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04\}$ for the low-dimensional setting, and $\alpha \in \{0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04, 0.08\}$ for the high-dimensional setting. Thus, for each randomly generated graph, we obtained 50 estimated CPDAGs from each algorithm, for each value of α.

All simulations were performed on an AMD Opteron(tm) Processor 6174 with 128 GB on Linux using R 2.15.1.

5.2 Estimation performance

For the low-dimensional setting, the two algorithms were basically indistinguishable in terms of estimation performance and computing time. For the high-dimensional setting, however, the performance of the algorithms was rather different. We therefore only discuss the high-dimensional results.

Figure 4 shows the estimation performance of the algorithms. The top-left panel shows that the PC-stable algorithm returns graphs with fewer edges than the PC algorithm, for all values of α. This is related to the fact that the PC-stable algorithm tends to perform more tests than the PC algorithm, see also Table 1.

The other panels show that the PC-stable algorithm has significantly better performance than the PC algorithm in terms of the number of estimation errors in the skeleton (top-right panel), the True Discovery Rate (TDR, bottom-left panel), and the Structural Hamming Distance (SHD, bottom-right panel), for all values of α. The TDR is defined as the proportion of edges in the estimated skeleton that are also present in the true skeleton. The SHD counts the minimum number of edge insertions, deletions, and flips that are needed in order to transform the estimated graph into the true one.

Figure 5 shows more detailed results for the estimated skeletons for one of the 250 graphs (randomly chosen), for four different values of α. We see that for each value of α shown, the original PC algorithm yielded a certain number of stable edges that were present for all

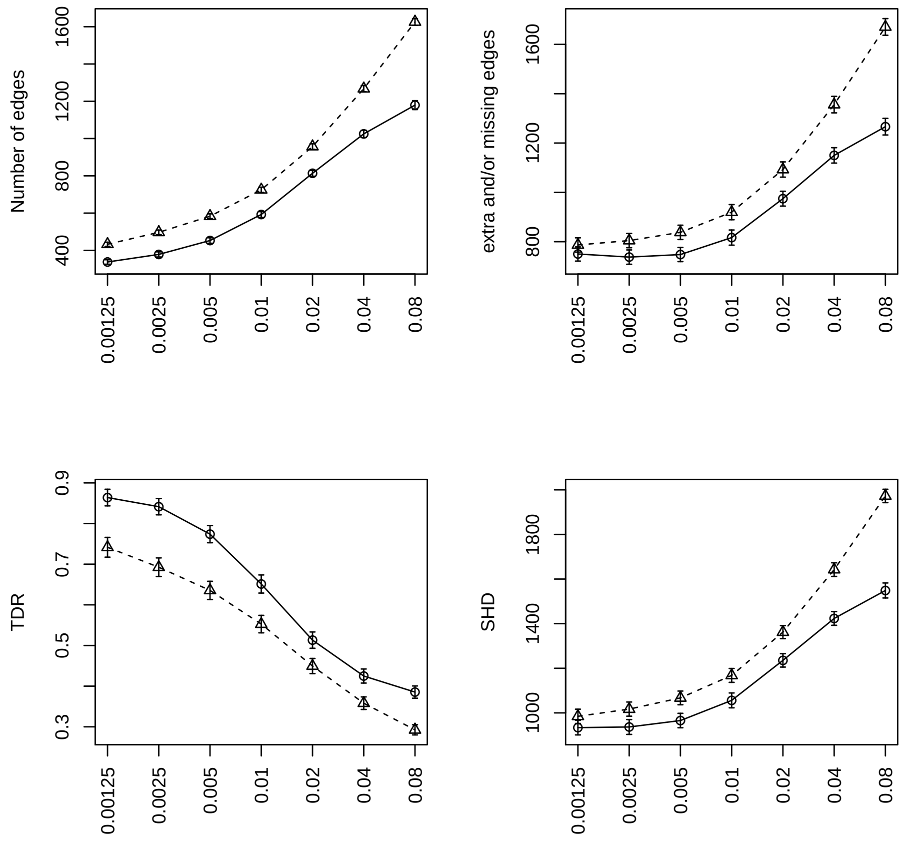
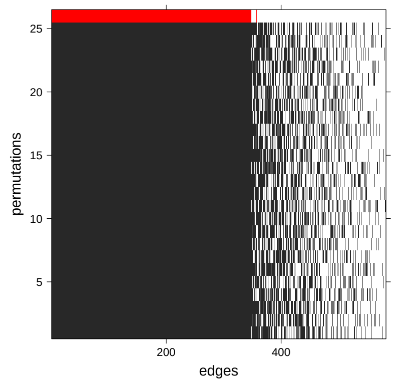
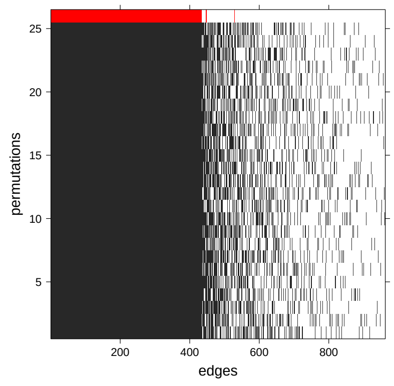


Figure 4: Estimation performance of the PC algorithm (triangles; dashed line) and the PC-stable algorithm (circles; solid line), for different values of α (x-axis displayed in $\log_2$ scale) in the high-dimensional simulation setting with $p = 1000$ and $n = 50$. The results are shown as averages plus or minus one standard deviation, computed over 250 randomly generated graphs and 50 random variable orderings per graph. (We note that PC-stable is order-independent with respect to the measures in the first three panels, but order-dependent with respect to the SHD, as this measure relies on the edge orientations.)

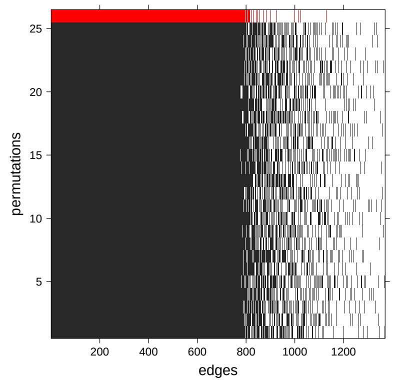
25 variable orderings, but also a large number of extra edges that pop in or out for different orderings. The PC-stable algorithm yielded far fewer edges (shown in red), and roughly captured the edges that were stable among the different variable orderings for the original PC algorithm.



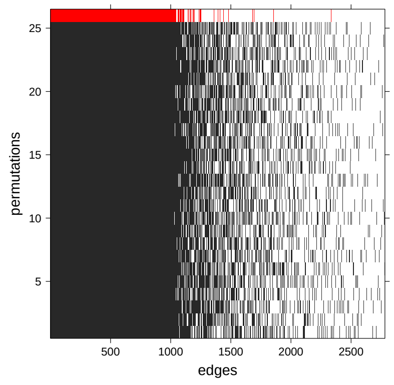
(a) $\alpha = 0.00125$



(b) $\alpha = 0.005$



(c) $\alpha = 0.02$



(d) $\alpha = 0.08$

Figure 5: Estimated edges with the PC algorithm (black) for 25 random orderings on the variables, as well as with the PC-stable algorithm (red), for a random graph from the high-dimensional setting with $p = 1000$ and $n = 50$ and four different values for α. (We used only 25 of the 50 random orderings in order to properly display the results.) The edges along the x-axes are ordered according to their presence in the 25 random orderings using the original PC algorithm. Edges that did not occur for any of the orderings were omitted.

5.3 Number of tests and computing time

One can easily deduce that Step 1 of the PC and PC-stable algorithms perform the same number of tests for $\ell = 0$, because the adjacency sets do not play a role at this stage. Moreover, for $\ell = 1$ the PC-stable algorithm performs at least as many tests as the PC algorithm, since the adjacency sets $\mathrm{a}(X_i)$ (see Algorithm 4.2) are always supersets of the adjacency sets $\mathrm{adj}(X_i)$ (see Algorithm 3.2). For larger values of ℓ, however, it is difficult to analyze the number of tests analytically.

Table 1 therefore shows the average number of tests that were performed by Step 1 of the two algorithms, separated by size of the conditioning set, where we considered the high-dimensional setting with $\alpha = 0.08$ since this was most computationally intensive. As expected the number of marginal correlation tests was identical for both algorithms. For $\ell = 1$ and $\ell = 2$, the PC-stable algorithm performed about twice as many tests as the PC algorithm, amounting to about 5.7×10^5 and 5.2×10^4 additional tests, respectively. For larger values of ℓ, the PC-stable algorithm performed fewer tests than the PC algorithm, since the additional tests for $\ell = 1$ and $\ell = 2$ lead to a sparser skeleton. However, since the PC algorithm also performed relatively few tests for larger values of ℓ, the absolute difference in the number of tests for large ℓ is rather small. In total, the PC-stable algorithm performed about 6.1×10^5 more tests than the original PC algorithm.

	PC algorithm	PC-stable algorithm
$\ell = 0$	5.40×10^5 (2.90×10^2)	5.40×10^5 (2.90×10^2)
$\ell = 1$	4.52×10^5 (6.98×10^3)	1.02×10^6 (1.45×10^4)
$\ell = 2$	6.70×10^4 (2.75×10^3)	1.19×10^5 (5.52×10^3)
$\ell = 3$	6.58×10^3 (4.83×10^2)	1.76×10^3 (2.62×10^2)
$\ell = 4$	1.08×10^3 (1.54×10^2)	2.07×10^2 (6.37×10^1)
$\ell = 5$	9.82×10^1 (3.72×10^1)	1.32×10^1 (1.35×10^1)
$\ell = 6$	3.23×10^0 (5.53×10^0)	0.36×10^0 (1.55×10^0)
$\ell = 7$	0.25×10^{-1} (0.45×10^0)	-
Total	1.07×10^6 (9.32×10^3)	1.68×10^6 (1.90×10^4)

Table 1: Number of tests performed by Step 1 of the PC and PC-stable algorithms for each size of the conditioning sets ℓ, in the high-dimensional setting with $p = 1000$, $n = 50$ and $\alpha = 0.08$. The results are shown as averages (standard deviations) over 250 random graphs and 50 random variable orderings per graph.

Table 2 shows the average runtime of the PC and PC-stable algorithms. We see that the PC-stable algorithm is somewhat slower than the PC algorithm for all values of α, which can be explained by the fact that the PC-stable algorithm tends to perform a larger number of tests (cf. Table 1).

6. Yeast gene expression data

We also compared the PC and PC-stable algorithms on the yeast gene expression data (Hughes et al., 2000) that were already briefly discussed in Section 1. In Section 6.1 we consider estimation of the skeleton of the CPDAG, and in Section 6.2 we consider estimation of bounds on causal effects.

We used the same pre-processed data as in Maathuis et al. (2010). These contain: (1) expression measurements of 5361 genes for 63 wild-type cultures (observational data of size 63×5361), and (2) expression measurements of the same 5361 genes for 234 single-gene deletion mutant strains (interventional data of size 234×5361).

	PC algorithm	PC-stable algorithm
$\alpha = 0.00125$	94.73 (3.58)	100.54 (3.84)
$\alpha = 0.025$	94.96 (2.51)	100.60 (2.70)
$\alpha = 0.05$	95.50 (3.14)	101.25 (3.33)
$\alpha = 0.01$	98.53 (3.60)	104.83 (3.47)
$\alpha = 0.02$	103.07 (3.44)	112.21 (3.71)
$\alpha = 0.04$	118.57 (4.54)	138.35 (5.24)
$\alpha = 0.08$	182.76 (7.42)	259.28 (11.04)

Table 2: Run time in seconds of the PC and PC-stable algorithms for the high-dimensional setting with $p = 1000$ and $n = 50$. The results are shown as averages (standard deviations) over 250 random graphs and 50 random variable orderings per graph.

6.1 Estimation of the skeleton

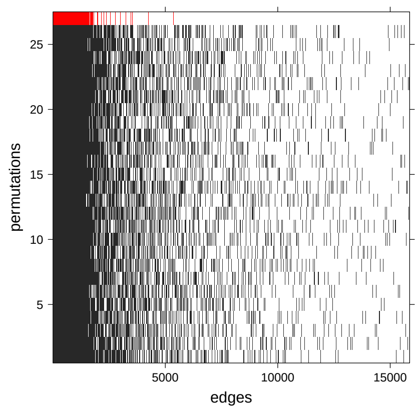
We applied the PC and PC-stable algorithms to the observational data. We saw in Section 1 that the PC algorithm yielded estimated skeletons that were highly dependent on the variable ordering, as shown in black in Figure 6 for the 26 variable orderings (the original ordering and 25 random permutations of the variables). The PC-stable algorithm does not suffer from this order-dependence, and the estimated skeleton is shown in the figure in red. We see that the PC-stable algorithm yielded a far sparser skeleton (2086 edges for PC-stable versus 5015-5159 edges for the PC algorithm, depending on the variable ordering). Just as in the simulations in Section 5 the PC-stable algorithm roughly captured the edges that were stable among the different variable orderings for the original PC algorithm.

To make “captured the edges that were stable” somewhat more precise, we defined the following two sets: Set 1 contained all edges (directed edges) that were present for all 26 variable orderings using the original PC algorithm, and Set 2 contained all edges (directed edges) that were present for at least 90% of the 26 variable orderings using the original PC algorithm. Set 1 contained 1478 edges (7 directed edges), while Set 2 contained 1700 edges (20 directed edges).

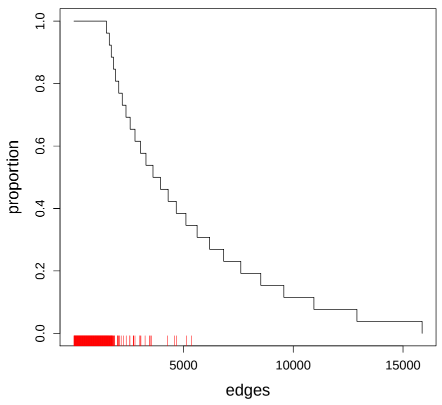
Table 3 shows how well the PC and PC-stable algorithms could find these stable edges in terms of true positives and false positives over the different variable orderings. We see that the number of true positives is about the same for both algorithms, while the PC-stable algorithm has far fewer false positives.

6.2 Estimation of causal effects

We used the interventional data as the gold standard for estimating the total causal effects of the 234 deleted genes on the remaining 5361 (see Maathuis et al. (2010)). We then defined the top 10% of the largest effects in absolute value as the target set of effects, and we evaluated how well IDA (Maathuis et al., 2009, 2010) identified these effects from the observational data.



(a) As Figure 1(a), plus the edges occurring in the estimated skeleton using the PC-stable algorithm (red, shown as permutation 27).



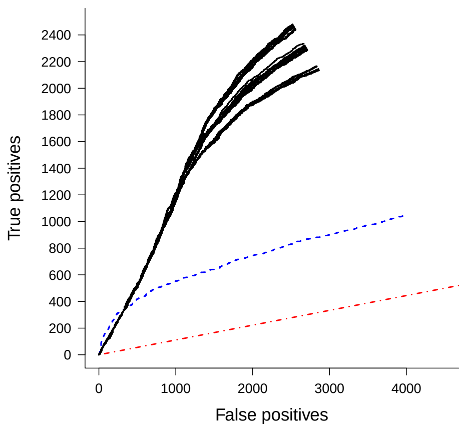
(b) The step function shows the proportion of the 26 permutations in which the edges were present for the original PC algorithm, where the edges are ordered as in Figure 6(a). The red bars show the edges present in the estimated skeleton using the PC-stable algorithm.

Figure 6: Analysis of estimated skeletons of the CPDAGs for the yeast gene expression data (Hughes et al., 2000), using the PC and PC-stable algorithms. The PC-stable algorithm yields an order-independent skeleton that roughly captures the edges that were stable among the different variable orderings for the original PC algorithm.

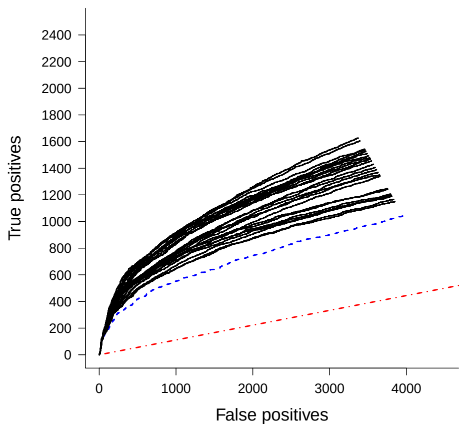
		Edges		Directed edges	
		PC algorithm	PC-stable algorithm	PC algorithm	PC-stable algorithm
Set 1	TP	1478 (0)	1478 (0)	7 (0)	7 (0)
	FP	3606 (38)	607 (0)	4786 (47)	1444 (2)
Set 2	TP	1688 (3)	1688 (0)	19 (1)	20 (0)
	FP	3396 (39)	397 (0)	4774 (47)	1431 (2)

Table 3: Number of true positives and false positives for edges and directed edges for the PC and PC-stable algorithms, using Sets 1 and 2 as target set. The results are shown as averages (standard deviations) over the 26 variable orderings.

We saw in Figure 1(b) that IDA with the original PC algorithm is highly order-dependent. Figure 7 shows the same analysis with PC-stable, using tuning parameters $\alpha = 0.01$ and $\alpha = 0.05$. We see that using PC-stable generally yielded better and more stable results than the original PC algorithm for both values of α, where $\alpha = 0.05$ seems to give best results for very small numbers of false positives, while $\alpha = 0.01$ gives best results for larger numbers of false positives. Note that the curves for $\alpha = 0.01$ are slightly worse than the reference curve of Maathuis et al. (2010) towards the beginning of the curves, but this can be explained by the fact that the original variable ordering seems to be especially "lucky" for this part of the curve (cf. Figure 1(b)). There is still some variability in the ROC curves in Figure 7 due to the order-dependent orientations in the PC-stable algorithm, but this variability is much less prominent than in Figure 1(b).



(a) ROC curves corresponding to the 25 random orderings of the variables (solid black), where the curves are generated as in Maathuis et al. (2010) but using PC-stable with $\alpha = 0.01$. The ROC curves from Maathuis et al. (2010) (dashed blue) and the one for random guessing (dashed-dotted red) are shown as references.



(b) ROC curves corresponding to the 25 random orderings of the variables (solid black), where the curves are generated as in Maathuis et al. (2010) but using PC-stable with $\alpha = 0.05$. The ROC curves from Maathuis et al. (2010) (dashed blue) and the one for random guessing (dashed-dotted red) are shown as references.

Figure 7: Analysis of yeast gene expression data (Hughes et al., 2000), for 25 random permutations of the variables, using the PC-stable algorithm. The resulting causal rankings are no longer highly order-dependent.

7. Discussion

Due to its computational efficiency, the PC algorithm is widely used for estimating CPDAGs in sparse high-dimensional settings. We found, however, that the order-dependence of the PC algorithm is especially problematic in these settings. We proposed a simple modification of the PC algorithm, called PC-stable, that removes a large part of this order-dependence and yields an order-independent skeleton. Existing high-dimensional consistency results for the PC algorithm remain valid for the PC-stable algorithm under the same conditions. Moreover, we showed that the PC-stable algorithm yields improved and more stable estimation in sparse high-dimensional settings, both for simulated data and a real data set. All software is implemented in the R-package `pcalg` (Kalisch et al., 2012).

Compared to the adaptation of Cano et al. (2008), the PC-stable algorithm is much simpler and preserves existing soundness, completeness, and high-dimensional consistency results. Moreover, PC-stable can be easily used together with other adaptations of the PC algorithm, for example hybrid versions of PC with score-based methods Singh and Valtorta (1993); Spirtes and Meek (1995); van Dijk et al. (2003) or the PC* algorithm (Spirtes et al., 2000, Section 5.4.2.3). Moreover, the modification in PC-stable is also relevant for the FCI algorithm (Spirtes et al., 2000) and the RFCI algorithm (Colombo et al., 2012), as the first step of these algorithms is identical to Step 1 of the PC algorithm. The existing consistency results of FCI and RFCI remain valid under the same conditions (Colombo et al., 2012) when using our modification.

The edge orientations generally remain order-dependent in the PC-stable algorithm. A possible solution for this problem would be to use stability selection (Meinshausen and Bühlmann, 2010) to find the most stable orientations, where the variable ordering should be permuted in each stability run. In fact, Stekhoven et al. (2012) already proposed a combination of IDA and stability selection which lead to improved performance when compared to plain IDA, but they used the original PC algorithm and did not permute the variable ordering. Our findings in this paper suggest that incorporating PC-stable into this approach and permuting the variable ordering in each stability run is likely to lead to further improvements.

References

S. A. Andersson, D. Madigan, and M. D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *Ann. Statist.*, 25(2):505–541, 1997.

A. Cano, M. Gómez-Olmedo, and S. Moral. A score based ranking of the edges for the PC algorithm. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models*, pages 41–48, 2008.

D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498, 2002.

D. Colombo, M.H. Maathuis, M. Kalisch, and T.S. Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *Ann. Statist.*, 40(1):294–321, 2012. ISSN 0090-5364.

G.F. Cooper and E.A. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

D. Dash and M.J. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth Conference on Uncertainty on Artificial Intelligence (UAI-99)*, pages 142–149. Morgan Kaufmann Publishers, Inc., 1999.

A. P. Dawid. Conditional independence for statistical operations. *Ann. Statist.*, 8:598–617, 1980.

N. Harris and M. Drton. PC algorithm for gaussian copula graphical models. *arXiv preprint arXiv:1207.0242*, 2012.

D. Heckerman, D. Geiger, and D.M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

T.R. Hughes, M.J. Marton, A.R. Jones, C.J. Roberts, R. Stoughton, C.D. Armour, H.A. Bennett, E. Coffey, H. Dai, Y.D. He, and et al. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, 2000.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *J. Mach. Learn. Res.*, 8:613–636, 2007.

M. Kalisch, B.A.G. Fellinghauer, E. Grill, M.H. Maathuis, U. Mansmann, P. Bühlmann, and G. Stucki. Understanding human functioning using graphical models. *BMC Medical Research Methodology*, 10(1):14, 2010.

M. Kalisch, M. Mächler, D. Colombo, M.H. Maathuis, and P. Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11): 1–26, 2012.

M.H. Maathuis, M. Kalisch, and P. Bühlmann. Estimating high-dimensional intervention effects from observational data. *Ann. Statist.*, 37(6A):3133–3164, 2009.

M.H. Maathuis, D. Colombo, M. Kalisch, and P. Bühlmann. Predicting causal effects in large-scale systems from observational data. *Nature Methods*, 7(4):247–248, 2010.

C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 403–411. Morgan Kaufmann Publishers, Inc., 1995.

N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

R. Nagarajan, S. Datta, M. Scutari, M. Beggs, G. Nolen, and C. Peterson. Functional relationships between genes associated with differentiation potential of aged myogenic progenitors. *Frontiers in Physiology*, 1(160), 2010.

J. Pearl. *Causality. Models, reasoning, and inference.* Cambridge University Press, Cambridge, 2000.

J. Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

M. Singh and M. Valtorta. An algorithm for the construction of bayesian network structures from data. In *Proceedings of the Ninth International Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pages 259–265. Morgan Kaufmann Publishers, Inc., 1993.

D.J. Spiegelhalter, A.P. Dawid, S.L. Lauritzen, and R.G. Cowell. Bayesian analysis in expert systems (with discussion). *Statistical Science*, 8:219–283, 1993.

P. Spirtes and C. Meek. Learning bayesian networks with discrete variables from data. In *Proceeding of the First International Conference on Knowledge Discovery and Data Mining*, pages 294–299, Menlo Park, CA: AAAI, 1995.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* MIT Press, Cambridge, second edition, 2000.

D.J. Stekhoven, I. Moraes, G. Sveinbjörnsson, L. Hennig, M.H. Maathuis, and P. Bühlmann. Causal stability ranking. *Bioinformatics*, 2012.

I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

S. van Dijk, L.C. can der Gaag, and D. Thierens. A skeleton-based approach to learning bayesian networks from data. In *Proceedings of the Seventh Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2003)*, pages 132–143. Springer Verlag, 2003.

X. Zhang, X.M. Zhao, K. He, L. Lu, Y. Cao, J. Liu, J.K. Hao, Z.P. Liu, and L. Chen. Inferring gene regulatory networks from gene expression data by pc-algorithm based on conditional mutual information. *Bioinformatics*, 2011.